

Proiectarea bazelor de date

Cuprins

I.	Proiectarea bazelor de date. Noțiuni introductive	2
1.	Date, informații, cunoștințe	2
2.	Colectarea și analizarea datelor. Modelul conceptual	2
3.	Entități. Instanțe. Atribute. Identificator unic.....	3
4.	Relații între entități	4
4.1.	Convenții de reprezentare a relațiilor.....	4
4.2.	Tipuri de relații	5
4.3.	Transferabilitate.....	5
4.4.	Tipuri și subtipuri	6
4.5.	Relații exclusive (arce)	6
4.6.	Relații ierarhice. Relații recursive	7
4.7.	Relații redundante si multiple.....	7
5.	Rezolvarea relațiilor many-to-many	8
6.	Modelarea datelor istorice	9
II.	Normalizarea datelor	10
1.	Ce este normalizarea?.....	10
2.	Prima formă normală.....	10
3.	A doua formă normală	11
4.	A treia formă normală.....	12

I. Proiectarea bazelor de date. Noțiuni introductive

1. Date, informații, cunoștințe

Auzim adesea vorbindu-se despre "Era informațiilor" sau "societate informațională" sau "tehnologia informației" însă de multe ori cuvântul "informație" este folosit fără a înțelege clar sensul acestui cuvânt, diferența dintre date, informații, cunoștințe.

În general, conținutul gândirii umane operează cu următoarele concepte:

1. **Date** – constau în material brut, fapte, simboluri, numere, cuvinte, poze fără un înțeles de sine stătător, neintegrate într-un context, fără relații cu alte date sau obiecte. Ele se pot obține în urma unor experimente, sondaje etc.
2. **Informații** – prin prelucrarea datelor și găsirea relațiilor dintre acestea se obțin informații care au un înțeles și sunt integrate într-un context. Datele organizate și prezentate într-un mod sistematic pentru a sublinia sensul acestor date devin informații. Pe scurt informațiile sunt date prelucrate. Informațiile se prezintă sub formă de rapoarte, statistici, diagrame etc.
3. **Cunoștințele** sunt colecții de date, informații, adevăruri și principii învățate, acumulate de-a lungul timpului. Informațiile despre un subiect reținute și înțelese și care pot fi folosite în luarea de decizii, formează judecăți și opinii devin cunoștințe. Cu alte cuvinte, cunoștințele apar în momentul utilizării informației.

2. Colectarea și analizarea datelor. Modelul conceptual

Primul pas în realizarea unei aplicații de baze de date este analiza datelor și realizarea unei **scheme conceptuale (model conceptual)** al acestor date.

În această etapă sunt analizate natura și modul de utilizare a datelor. Sunt identificate datele care vor trebui memorate și procesate, se împart aceste date în grupuri logice și se identifică relațiile care există între aceste grupuri.

Analiza datelor este un proces uneori dificil, care necesită mult timp, însă este o etapă absolut obligatorie. Fără o analiză atentă a datelor și a modului de utilizare a acestora, vom realiza o bază de date care putem constata în final că nu întrunește cerințele beneficiarului. Costurile modificării acestei baze de date este mult mai mare decât costurile pe care le-ar fi implicat etapa de analiză și realizare a modelului conceptual. Modificarea modelului conceptual este mult mai ușoară decât modificarea unor tabele deja existente, care eventual conțin și o mulțime de date. Ideea de bază a analizei datelor și construirii modelului conceptual este "să măsoari de două ori și să tai o singură dată".

Informațiile necesare realizării modelului conceptual se obțin folosind metode convenționale precum interviuarea oamenilor din cadrul organizației și studierea documentelor folosite.

Odată obținute aceste informații ele trebuiesc reprezentate într-o formă convențională care să poată fi ușor înțeleasă de toată lumea. O astfel de reprezentare este **diagrama entități-relații**, numită și **harta relațiilor**, sau **ERD-ul** (Entity Relationship Diagram). Aceste scheme sunt un instrument util care ușurează comunicarea dintre specialiștii care proiectează bazele de date și programatorii pe de o parte și beneficiarii, pe de altă parte. Aceștia din urmă pot înțelege cu ușurință o astfel de schemă, chiar dacă nu sunt cunoscători în domeniul IT.

În concluzie putem sublinia câteva caracteristici ale ERD-urilor:

- **sunt un instrument de proiectare**
- **sunt o reprezentare grafică a unui sistem de date**
- **oferă un model conceptual de înalt nivel al bazelor de date**
- **sprijină înțelegerea de către utilizatori a datelor și a relațiilor dintre acestea**
- **sunt independente de implementare.**

În cele ce urmează vom prezenta principalele elemente care intră în componența unui ERD precum și convențiile de reprezentare a acestora.

3. Entități. Instanțe. Atribute. Identificator unic.

O *entitate* este un lucru, obiect, persoană sau eveniment care are semnificație pentru afacerea modelată, despre care trebuie să colectăm și să memorăm date. O entitate poate fi un lucru real, tangibil precum o clădire, o persoană, poate fi o activitate precum o programare sau o operație, sau poate fi o noțiune abstractă.

O entitate este reprezentată în ERD printr-un dreptunghi cu colțurile rotunjite. Numele entității este întotdeauna un *substantiv la singular* și se scrie în partea de sus a dreptunghiului cu *majuscule*, ca în figura 1.

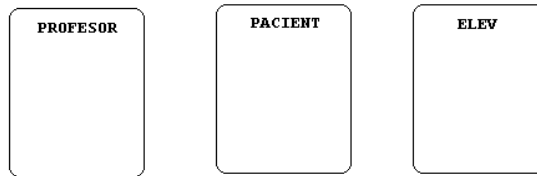


Figura 1- Exemple de entități și modul de reprezentare

O entitate este de fapt o clasă de obiecte și pentru orice entitate există mai multe *instanțe* ale sale. O instanță a unei entități este un obiect, persoană, eveniment, particular din clasa de obiecte care formează entitatea. De exemplu, elevul **Popescu** din clasa a IX-a A de la Liceul de Informatică din localitatea **București** este o instanță a entității **ELEV**.

După cum se vede pentru a preciza o instanță a unei entități, trebuie să specificăm unele caracteristici ale acestui obiect, să-l descriem (precizăm de exemplu numele, clasa, școala etc). Așadar, după ce am identificat entitățile trebuie să descriem aceste entități în termeni reali, adică să le stabilim *atributele*. Un atribut este orice detaliu care servește la identificarea, clasificarea, cuantificarea, sau exprimarea stării unei instanțe a unei entități. Atributele sunt informații specifice ce trebuie cunoscute și memorate.

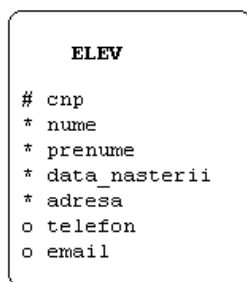


Figura 2- Entitatea ELEV

De exemplu atributele entității **ELEV** sunt nume, prenume, adresa, număr de telefon, adresa de email, data nașterii etc.

În cadrul unui ERD, atributele se vor scrie imediat sub numele entității, cu litere mici. Un atribut este un *substantiv la singular* (vezi figura 2).

Un atribut poate fi *obligatoriu* sau *opțional*. Dacă un atribut este obligatoriu, pentru fiecare instanță a entității respective trebuie să avem o valoare pentru acel atribut, de exemplu este obligatoriu să cunoaștem numele elevilor. Pentru un atribut opțional putem avea instanțe pentru care nu cunoaștem valoarea atributului respectiv. De exemplu atributul **email** al entității **ELEV** este opțional, un elev putând să nu aibă adresă de email. Un atribut obligatoriu este precedat în ERD de un asterisc *, iar un atribut opțional va fi precedat de un cerculeț o.



Figura 3- Entitatea CARTE

Atributele care definesc în mod unic instanțele unei entități se numesc *identificator unic* (**UID** **U**nique **ID**entifier). UID-ul unei entități poate fi compus dintr-un singur atribut, de exemplu codul numeric personal poate fi un identificator unic pentru entitatea **ELEV**.

În alte situații, identificatorul unic este compus dintr-o combinație de două sau mai multe atribute (**UID compus**). De exemplu combinația dintre titlu, numele autorului și data apariției poate forma unicul identificator al entității **CARTE**. Oare combinația titlu și nume autor nu era suficientă? Răspunsul este NU, deoarece pot exista de exemplu mai multe volume scrise de Mihai Eminescu având toate titlul Poezii, dar apărute la date diferite.

Dacă se recurge la o modalitate de identificare printr-un cod artificial oferit în mod automat de program, e vorba de **UID artificial**.

Atributele care fac parte din identificatorul unic al unei entități vor fi precedate de semnul diez # (figura 2 și 3). *Atributele din UID sunt întotdeauna obligatorii*, însă semnul # este suficient, nu mai trebuie pus și un semn asterisc în fața acestor atribute.

Valorile unor atribute se pot modifica foarte des, ca de exemplu atributul vârstă. Spunem în acest caz că avem de a face cu un *atribut volatil*. Dacă valoarea unui atribut însă se modifică foarte rar sau deloc (de exemplu data nașterii) acesta este un atribut *non-volatil*. Evident este de preferat să folosim atribute non-volatile atunci când acest lucru este posibil.

4. Relații între entități

În lumea reală, obiectele nu există izolat. Între ele există relații. Așadar, după ce ai identificat care sunt entitățile și atributele acestor entități este timpul să punem în evidență relațiile care există între aceste entități, modul în care acestea comunică între ele. O **relație** este o asociere, legătură, sau conexiune existentă între entități și care are o semnificație pentru afacerea modelată. Orice relație este bidirecțională, legând două entități sau o entitate cu ea însăși. De exemplu, elevii studiază mai multe materii, o materie e studiată de către elevi.

Orice relație este caracterizată de următoarele elemente:

1. **numele relației** ;
2. **opționalitatea relației**;
3. **gradul (cardinalitatea) relației**.

Să luăm de exemplu relația existentă între entitățile **JUCĂTOR** și **ECHIPĂ**. Vom spune:

Un **JUCĂTOR** joacă într-o **ECHIPĂ**. Și la o **ECHIPĂ** trebuie să joace unul sau mai mulți **JUCĂTORI**.

- **Numele relației** este: *joacă*.
- Pentru a stabili **opționalitatea** relației trebuie să răspundem la următoarea întrebare: Un jucător **trebuie** să joace într-o echipă? Se poate ca un jucător să nu joace în nici o echipă? Dacă acceptăm că toți jucătorii trebuie să joace într-o echipă relația este obligatorie sau mandatorie și vom spune: Un **JUCĂTOR trebuie** să joace într-o **ECHIPĂ**. Dacă însă acceptăm că există jucători care nu joacă în nici o echipă (de exemplu li s-a terminat contractul și în momentul de față nu mai joacă la nici o echipă), atunci relația este opțională. În acest caz vom spune: Un **JUCĂTOR poate** juca la o **ECHIPĂ**.
- **Cardinalitatea** relației este dată de numărul de instanțe ale entității din partea dreaptă a relației care pot intra în relație cu o instanță a entității din partea stângă a relației. Adică va trebui să răspundem la întrebări de genul: La câte echipe poate juca un jucător? Răspunsurile posibile sunt **unul și numai unul**, sau **unul sau mai mulți**. Vom spune:

Un **JUCĂTOR** trebuie/poate să joace la o **ECHIPĂ și numai una**.

sau Un **JUCĂTOR** trebuie/poate să joace la **una sau mai multe ECHIPĂ**.

Cea mai realistă variantă a relației este așadar: Un **JUCĂTOR** poate să joace la o **ECHIPĂ** și numai una.

4.1. Convenții de reprezentare a relațiilor

În cadrul diagramei entități-relații, o relație va fi reprezentată printr-o linie ce unește cele două entități.

Deoarece o relație este bidirecțională, linia ce unește cele două entități este compusă din două segmente distincte, câte una pentru fiecare entitate. Tipul segmentului ce pleacă de la o entitate ne va indica **opționalitatea** relației dintre această entitate și entitatea aflată în cealaltă parte a relației. Dacă acest segment este continuu este vorba de o relație obligatorie, o linie întreruptă indică o relație opțională.

De exemplu în figura 4 segmentul ce pleacă de la entitatea **JUCĂTOR** fiind **întreruptă** înseamnă că un jucător **poate** juca la o echipă, adică relația este opțională. Segmentul ce pleacă dinspre entitatea **ECHIPĂ** este **continuu**, deci la o echipă **trebuie** să joace jucători.

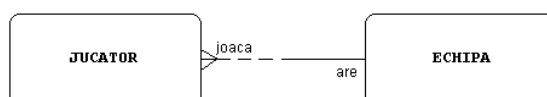


Figura 4 – Reprezentarea relațiilor

Modul în care o linie se termină spre o entitate este important. Dacă se termină printr-o linie simplă, înseamnă că o instanță și numai una a acestei entități este în relație cu o instanță a celeilalte entități. În exemplul anterior, linia de la **JUCĂTOR** la **ECHIPĂ** se termină în partea dinspre **ECHIPĂ** cu o **linie simplă**, deci un jucător joacă la o echipă **și numai una**.

Dacă linia se termină cu trei linii (picior de cioară) înseamnă că mai multe instanțe ale entității pot corespunde unei instanțe a celeilalte entități. În exemplul anterior linia de la **ECHIPĂ** la **JUCĂTOR** se termină cu **piciorul de cioară**, înseamnă că unei instanțe a entității **ECHIPĂ** îi corespund mai multe instanțe ale entității **JUCĂTOR**, adică o echipă are **unul sau mai mulți** jucători.

Caracteristica relației	Valoare	Mod de reprezentare
Numele relației	un verb	se scrie deasupra relației
Opționalitatea	relație obligatorie (TREBUIE)	linie continuă ———
	relație opțională (POATE)	linie întreruptă - - - - -
Cardinalitatea	una și numai una	linie simplă —
	una sau mai multe	picior de cioară ≡

4.2. Tipuri de relații

Variantele de relații ce pot exista între două entități sunt prezentate mai jos:

- **relații one-to-one** – acest tip de relație este destul de rar întâlnit. Uneori astfel de relații pot fi modelate transformând una dintre entități în atribut al celeilalte entități.

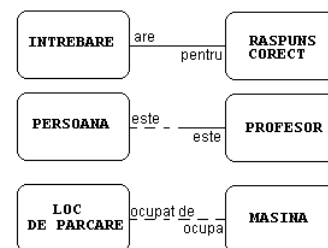


Figura 5 Relații one-to-one

- **relații one-to-many** – sunt cele mai întâlnite tipuri de relații, însă și aici cazurile c și d prezentate în figura 6 sunt mai puțin uzuale.

Să facem câteva observații pe marginea exemplurilor din figura 6.

Cazul a este foarte des întâlnit. La **cazul b**, am ales o relație opțională dinspre **POEZIE** spre **POET** deoarece poate fi vorba de o poezie populară și în acest caz nu există un poet cunoscut. La **cazul c**, am considerat că o formație nu poate exista fără a avea cel puțin un membru, însă un artist poate avea o carieră solo, deci nu face parte din nici o formație. **Varianta d** modelează o colecție de filme memorate pe CD-uri. Pentru afacerea considerată, un CD conține obligatoriu un film, dar unul singur, însă un film poate să nu încapă pe un singur CD de aceea el este poate fi memorat pe unul sau mai multe CD-uri.

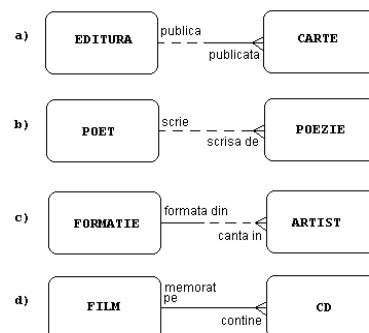


Figura 6 Relații one-to-many

- **relații many-to-many** – aceste tipuri de relații apar în prima fază a proiectării bazei de date, însă ele trebuie să fie ulterior eliminate. Figura I.1.7 prezintă câteva exemple de relații many-to-many. La punctul b am considerat că un curs poate apărea pe oferta de cursuri a unei facultăți, însă poate să nu fie aleasă de nici un student de aceea un curs **poate** fi urmat de unul sau mai mulți studenți. Invers, este posibil ca un student să fi terminat studiile și să se pregătească pentru susținerea examenului de licență și de aceea el nu mai frecventează nici un curs. La punctul c, un profesor angajat al unei școli **trebuie** să predea cel puțin o disciplină. Iar o disciplină din planul de învățământ trebuie să fie predată de cel puțin un profesor.

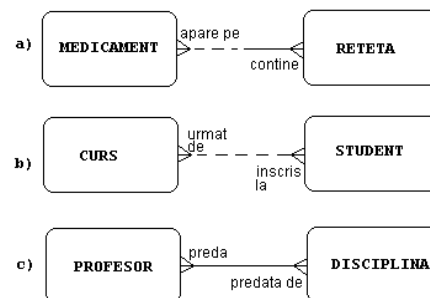


Figura 7 Relații many-to-many

4.3. Transferabilitate

Spunem că o relație este nontransferabilă dacă o asociație între două instanțe ale celor două entități, odată stabilită, nu mai poate fi modificată. Nontransferabilitatea unei relații se reduce la faptul că valorile cheii străine corespunzătoare relației respective nu pot fi modificate. Condiția de nontransferabilitate a unei relații este asigurată prin program. De aceea trebuie să documentăm această restricție. În ERD o relație nontransferabilă se notează cu un romb pe linia corespunzătoare relației, înspre entitatea a cărei cheie străină nu este permis să o modificăm (adică în partea cu many a unei relații one-to-many).

În figura 8 este dat un exemplu de relație nontransferabilă. Este vorba despre notele date elevilor. Este normal ca o notă dată unui elev să nu poată fi apoi transferată unui alt elev.

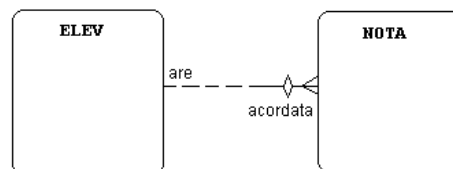


Figura 8 Relații nontransferabile

4.4. Tipuri și subtipuri

În lumea reală obiectele sunt de obicei clasificate. Astfel vorbim despre animale vertebrate și nevertebrate, despre licee teoretice, colegii, grupuri școlare etc. E normal ca în modelarea bazelor de date să putem modela și astfel de clasificări.

Un **subtip** sau o **subentitate** este o clasificare a unei entități care are caracteristici comune cu entitatea generală, precum atribute și relații. Subtipurile se reprezintă în cadrul hărții relațiilor ca entități în interiorul altei entități. Atributele și relațiile comune tuturor subtipurilor se vor reprezenta la nivelul **supertipului**, sau **superentității**. Atributele și relațiile supertipului vor fi moștenite de către subtipuri.

Un subtip poate avea la rândul său alte subtipuri incluse.

Folosirea subtipurilor și supertipurilor

Subtipurile trebuie să respecte două reguli importante:

- trebuie să acopere toate cazurile posibile de instanțe ale supertipului, cu alte cuvinte, orice instanță a supertipului trebuie să aparțină unui subtip. De multe ori ERD-urile includ un subtip "ALTUL" pentru a acoperi toate situațiile, și pentru a permite viitoare dezvoltări ale modelului.
- subtipurile trebuie să se excludă reciproc. Această regulă se traduce pe exemplul de mai sus în faptul că un angajat nu poate fi, de exemplu, și manager și secretară în același timp.

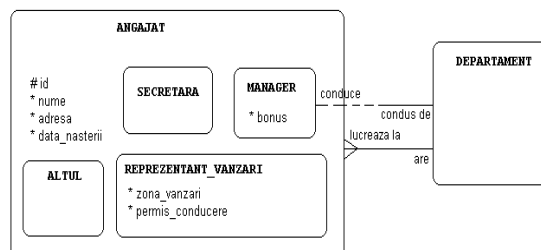


Figura 9

4.5. Relații exclusive (arce)

În unele situații, relațiile se pot exclude reciproc, adică dintr-un grup de relații, la un moment dat doar una dintre ele poate avea loc. De exemplu, un cont anume la o bancă este deținut fie de o persoană fizică fie de o firmă dar nu de ambele tipuri de clienți simultan. Un grup de relații exclusive este reprezentat în harta relațiilor printr-un arc peste relațiile care fac parte din respectivul grup. Toate relațiile ce fac parte din grupul de relații exclusive trebuie să aibă aceeași opționalitate. Un arc aparține unei singure entități, adică va include doar relații care pleacă de la o aceeași entitate.

O entitate poate avea mai multe arce, dar o anumită relație nu poate face parte decât dintr-un singur arc.

Există două tipuri de relații exclusive:

- relații exclusive **obligatorii** în care toate relațiile ce fac parte din arcul respectiv sunt obligatorii, ceea ce înseamnă că de fiecare dată, una dintre relații are obligatoriu loc. Este și cazul din figura 10. Evident că un cont trebuie să fie deținut de o persoană fizică sau de o firmă, o a treia variantă neexistând.
- relații exclusive **opționale** caz în care toate relațiile ce fac parte din arc sunt opționale. În acest caz de fiecare dată are loc cel mult una dintre relații, existând varianta ca pentru o instanță a entității căreia aparține arcul să nu aibă loc nici una din relațiile din grupul respectiv. În figura 11, este exemplificată situația în care un elev poate opta să facă parte din echipa de fotbal, sau să participe la cercul literar sau la cercul de informatică. Însă regulile școlii prevăd ca un elev să nu participe la două astfel de activități extrașcolare. Relațiile fiind opționale, înseamnă că un elev are libertatea de a decide să nu participe la nici o activitate extrașcolară.

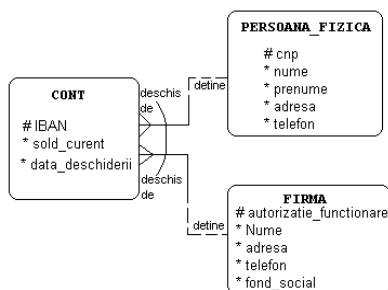


Figura 10 Relații exclusive obligatorii

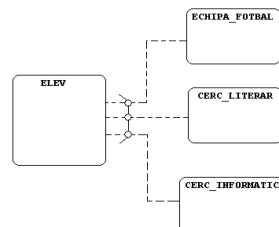


Figura 11 Relații exclusive opționale

4.6. Relații ierarhice. Relații recursive

Să analizăm care este structura personalului într-o firmă oarecare. În figura 12 este prezentată doar o parte din organigrama unei firme.

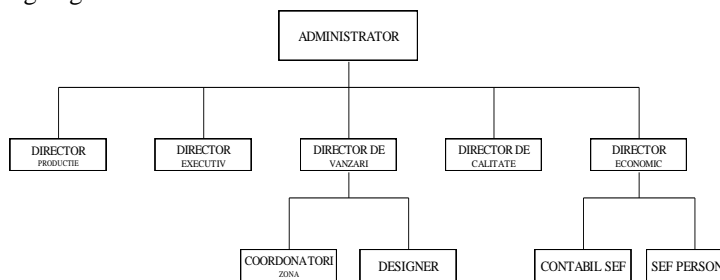


Figura 12 Organigrama unei firme

Un model de proiectare a unei astfel de structuri într-o bază de date ar fi cea din figura 13.

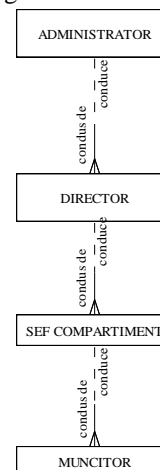


Figura 13 Implementarea unei structuri ierarhice

Problema este că fiecare tip de angajat din figura anterioară este de fapt un angajat și probabil există foarte multe atribute comune tuturor acestor entități ca de exemplu nume, prenume, adresă, telefon, email, data nașterii etc. Vom putea de aceea modela această structură cu ajutorul unei singure entități numită **ANGAJAT**. Însă fiecare angajat poate fi condus de către un alt angajat. Așadar vom avea o relație de la entitatea **ANGAJAT** la ea însăși. O astfel de relație se numește **relație recursivă**.

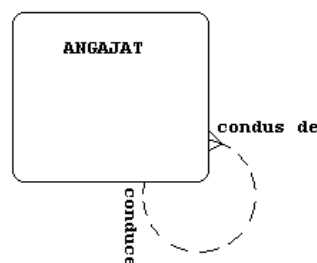


Figura 14 Implementarea unei structuri ierarhice folosind relații recursive

4.7. Relații redundante si multiple

Atunci când o relație poate fi dedusă din alte relații spunem că acea relație este redundantă (figura 15). Se observă că un elev face parte dintr-o clasă, iar la aceea clasă predau mai mulți profesori. Așadar relația Profesor și Elev nu mai este necesară deoarece putem deduce profesorii care îi predau unui elev, aflând profesorii clasei din care face parte elevul. Această relație poate fi eliminată (figura 16).

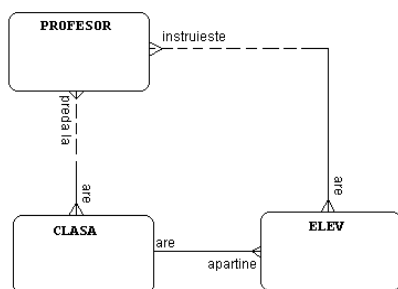


Figura 15

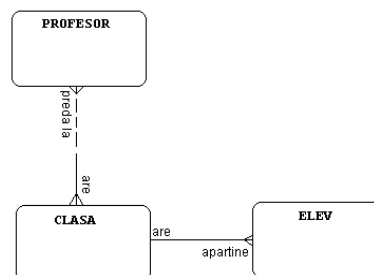
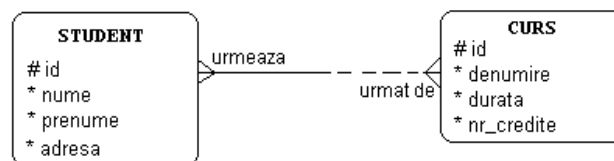


Figura 16

5. Rezolvarea relațiilor many-to-many

După cum am precizat mai devreme relațiile many-to-many pot apărea într-o primă fază a proiectării bazei de date însă ele nu au voie să apară în schema finală. Să considerăm relația din figura 17 dintre entitățile STUDENT și CURS. Se știe că orice curs se termină în general cu un examen. Unde vom memora nota studentului la fiecare examen?

Figura 17



Dacă încercăm să introducem atributul **NOTA** la entitatea **STUDENT**, nu vom ști cărei materii corespunde acea notă, întrucât unei instanțe a entității student îi corespund mai multe instanțe ale entității **CURS**. Invers dacă încercăm să memorăm nota în cadrul entității **CURS**, nu vom ști cui student îi aparține acea notă.

Rezolvarea unei relații many-to-many constă introducerea unei noi entități numită *entitate de intersecție*, pe care o legăm de entitățile originale prin câte o relație one-to-many.

Pașii în rezolvarea unei relații many-to-many sunt următorii:

- 1) Se găsește entitatea de intersecție, pentru exemplul nostru vom introduce entitate **INSCRIERE**.
- 2) Crearea noilor relații
 - opționalitatea: relațiile care pleacă **din entitatea de intersecție sunt întotdeauna obligatorii în această parte**. În partea dinspre entitățile originale, relațiile vor păstra opționalitatea relațiilor inițiale.
 - cardinalitatea: ambele relații sunt de tip one-to-many, iar partea cu many va fi întotdeauna înspre entitatea de intersecție.
 - numele noilor relații
- 3) Adăugarea de atribute în cadrul entității de intersecție, dacă acestea există. În exemplul nostru ne poate interesa de exemplu data la care s-a înscris un student la un curs, data la care a finalizat cursul precum și nota obținută la sfârșitul cursului.
- 4) Stabilirea identificatorului unic pentru entitatea de intersecție: dacă entitatea de intersecție nu are un identificator unic propriu, atunci acesta se poate forma din identificatorii unici ai entităților inițiale la care putem adăuga atribute ale entității de intersecție.
În exemplul nostru, identificatorul unic al entității de intersecție este format din id-ul studentului, id-ul cursului și data înscrierii la curs.
- 5) Faptul că identificatorul unic al unei entități preia identificatorul unic din altă entitate cu care este legată este reprezentat grafic prin bararea relației respective, înspre entitatea care preia UID-ul celeilalte entități.

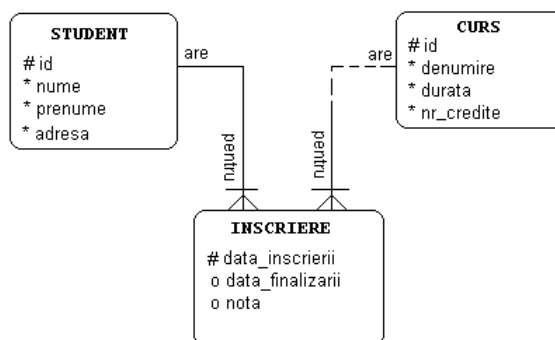


Figura 18

6. Modelarea datelor istorice

Viața înseamnă schimbare, orice lucru se schimbă de-a lungul timpului, și nu doar obiectele se modifică în timp dar chiar și relațiile dintre aceste obiecte se schimbă. Prețul produselor poate suferi modificări destul de des. Factorii care duc la aceste modificări pot fi dintre cei mai diverși, rata inflației, anotimpul etc. Așadar atributul **preț** din cadrul entității **produs** se modifică de-a lungul timpului. Dacă nu ne interesează decât prețul actual al fiecărui produs modelul este foarte simplu, ca cel din figura 19.

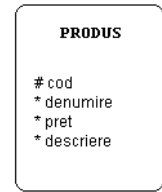


Figura 19

Dacă însă pentru afacerea modelată este important să reținem un istoric al prețurilor pentru fiecare produs, atunci atributul preț se va transforma într-o nouă entitate

Atributul **data_sfarsit** este opțional, deoarece data până la care este valabil prețul curent al unui produs nu este de obicei cunoscut.

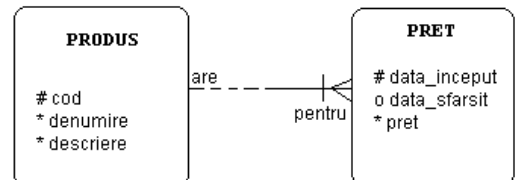
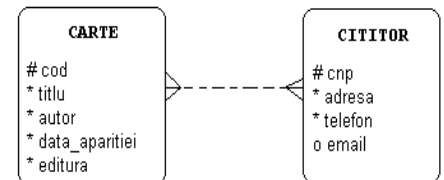


Figura 20

Vom considera acum o situație puțin mai dificilă. Să presupunem că dorim să modelăm o bază de date pentru o bibliotecă. Evident este important de reținut un istoric al tuturor împrumuturilor, deoarece pe baza acestora, se pot afla domeniile de interes ale cititorilor, și astfel vom ști ce achiziții de carte să facem în viitor, vom putea determina uzura cărților astfel încât să le putem înlocui etc.

Într-o primă fază vom obține o relație de many-to-many între entitățile **CARTE** și **CITITOR**. Fiecare carte poate fi împrumutată de mai mulți cititori (evident nu în același timp), și fiecare cititor poate împrumuta mai multe cărți.

Figura 21



Rezolvăm această relație many-to-many.

Să verificăm că acest caz este cel corect. Cheia primară este acum combinația coloanelor **cod_carte** și **data_imprumut**. Poate un cititor împrumuta două cărți în aceeași dată? Adică următoarele două înregistrări pot exista simultan în tabela **ISTORIC_IMPRUMUTURI**? Răspunsul este DA, combinația celor două coloane, pentru cele două înregistrări fiind unică.

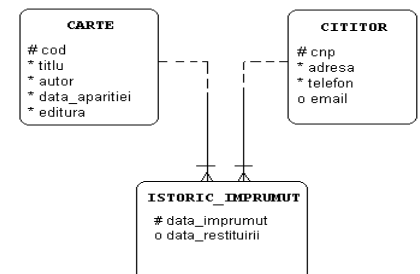


Figura 22

Deci bararea automată a celor două relații dinspre entitatea de intersecție nu este întotdeauna o soluție corectă. Pentru a evita aceste complicații putem recurge la introducerea unei chei artificiale în entitatea de intersecție. În exemplul nostru se poate decide ca pentru fiecare împrumut în parte să se completeze câte o fișă separată care are un număr unic. Obținem modelul din figura 23, care este de asemenea unul corect.

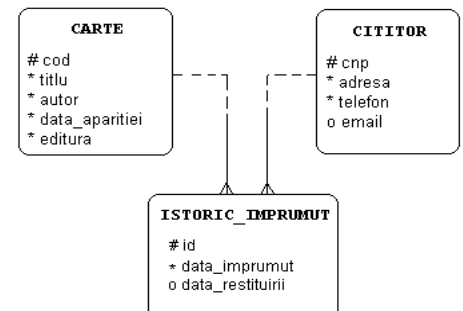


Figura 23 - Introducerea unei chei artificiale

II. Normalizarea datelor

1. Ce este normalizarea?

Normalizarea este o tehnică de proiectare a bazelor de date prin care se elimină (sau se evită) anumite anomalii și inconsistențe a datelor. O baza de date bine proiectată nu permite astfel ca datele să fie redundante, adică aceeași informație să se găsească în locuri diferite, sau să memorezi în baza de date, informații care se pot deduce pe baza altor informații memorate în aceeași bază de date.

Anomaliile care pot să apară la o bază de date nenormalizată sunt următoarele:

- **anomalii la actualizarea** datelor la o bibliotecă se înregistrează într-o tabelă următoarele date despre cărți: ISBN, titlu, autor, preț, subiect, editura, adresa editurii. La un moment dat o editură își schimbă adresa. Bibliotecara va trebui să modifice adresa editurii respective, în înregistrările corespunzătoare tuturor cărților din bibliotecă apărute la respectiva editură. Dacă această modificare nu se face cu succes, unele dintre înregistrări rămânând cu vechea adresă, apare din nou o inconsistență a datelor.
- **anomalii de inserare** – în exemplul anterior, nu vom putea memora adresa unei edituri, lucru inacceptabil dacă dorim să avem informații și despre edituri a căror cărți nu le avem în bibliotecă, eventual de la care dorim să facem comenzi.
- **anomalii de ștergere** – să presupunem că într-o tabelă memorăm următoarele informații: codul studentului, codul cursului, codul profesorului. La un moment dat, nici un student nu mai dorește să participe la un anumit curs. Ștergând toate înregistrările corespunzătoare cursului, nu vom mai putea ști niciodată cine predă acel curs.

Edgar Codd a definit primele trei forme normale 1NF, 2NF și 3NF. Ulterior s-au mai definit formele normale 4NF, 5NF, 6NF care însă sunt rar folosite în proiectarea bazelor de date.

2. Prima formă normală

O entitate se găsește în prima formă normală dacă și numai dacă:

- nu există attribute cu valori multiple;
- nu există attribute sau grupuri de attribute care se repetă.

Cu alte cuvinte toate attributele trebuie să fie atomice, adică să conțină o singură informație.

Dacă un atribut are valori multiple, sau un grup de attribute se repetă, atunci trebuie să creai o entitate suplimentară pe care să o leați de entitatea originală printr-o relație de 1:m. În noua entitate vor fi introduse attributele sau grupurile de attribute care se repetă.

Să considerăm entitatea din figura 24, referitoare la notele elevilor unei clase. Câteva observații referitoare la această entitate: câte discipline are un elev? Câte perechi (disciplina, nota) va trebui să aibă entitatea Elevi? Să spunem că știm exact câte discipline maxim poate studia un elev. Ce se întâmplă dacă în anul viitor școlar acest număr de discipline va fi mai mare? În plus, la o materie un elev poate avea mai multe note. Câte note? Cum memorăm aceste note? Le punem în câmpul corespunzător disciplinei cu virgulă între ele?

Cum rezolvăm această problemă? Vom crea o nouă entitate în care vom introduce disciplina și nota la disciplina respectivă (figura 25).

În acest fel fiecărui elev îi pot corespunde oricâte note, iar la o disciplină poate avea oricâte note, singura restricție conform acestui model fiind că un elev nu va putea primi în aceeași zi la aceeași materie mai multe note.

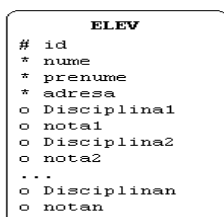


Figura 24

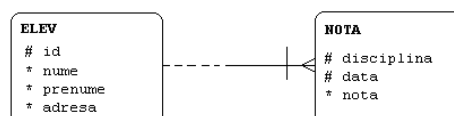


Figura 25

Un alt exemplu de încălcare a regulilor primei forme normale, puțin mai "ascuns", este prezentat în figura 26. De ce? Pentru că adresa este de forma "str. Florilor, bl. 45, sc. A, ap. 28, etaj 3, Brașov, cod 123123", formă care de fapt conține mai multe informații elementare. Așadar, în mod normal acest atribut ar trebui "spart" în mai multe attribute ca în figura 27.

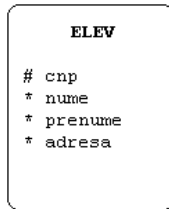


Figura 26

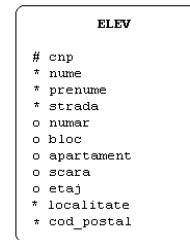


Figura 27

Noile attributele introduse sunt opționale întrucât dacă elevul locuiește la casă, probabil attributele bloc, apartament, scara, etaj, nu au sens. Invers dacă elevul locuiește la bloc, probabil nu poate fi completat numărul.

Pentru acest tip de încălcare a regulilor forme normale 1NF poate fi totuși ignorată, decizia depinzând de natura fenomenului, sau afacerii modelate. În exemplul anterior, întrucât datele din interiorul unei adrese este puțin probabil să se modifice, modificându-se el mult adresa completă a unui elev, se poate decide să nu operăm modificarea anterioară. Dacă însă aceste informații s-ar modifica frecvent, de exemplu denumirile străzilor s-ar modifica mereu, atunci probabil modificarea este de dorit.

3. A doua formă normală

O entitate se găsește în a doua formă normală dacă și numai dacă se găsește în prima formă normală și în plus orice atribut care nu face parte din UID (unique identifier) va depinde de întregul UID nu doar de o parte a acestuia.

De exemplu, memorăm angajații unui departament într-o entitate ca cea din figura 28.

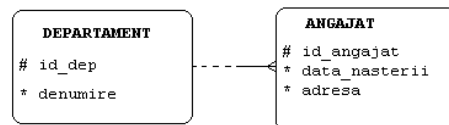
Se observă că **data_nasterii** și **adresa** sunt două attribute care depind doar de id-ul angajatului nu de întregul UID care este combinația dintre attributele **id_dep** și **id_angajat**.

Figura 28



Această situație se rezolvă prin crearea unei noi entități **ANGAJAT**, pe care o legăm de entitatea **DEPARTAMENT** printr-o relație **1:m**.

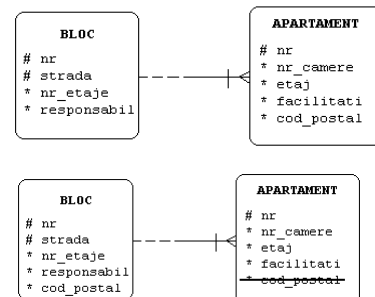
Figura 29



O situație mai specială este în cazul relațiilor barate, când trebuie ținut seama că UID-ul unei entități este compus din attribute din entitatea respectivă plus un atribut sau mai multe attribute provenite din relația barată.

Să considerăm următorul exemplul din figura 30.

Se observă că UID-ul entității **APARTAMENT** este compus din combinația a trei attribute: numărul apartamentului, numărul blocului și strada. Deci toate attributele din entitatea **APARTAMENT** care nu fac parte din UID, trebuie să depindă de întregul UID. Dar se știe că atributul **cod_postal** depinde doar de strada și de numărul blocului, nu și de numărul apartamentului. Acest lucru ne spune că acest atribut nu este memorat la locul potrivit. Deoarece depinde doar de combinația (strada, nr_bloc), înseamnă că de fapt depinde de UID-ul entității **bloc**. Așadar vom muta atributul **cod_postal** în entitatea **BLOC**.



Observație. Dacă o entitate se găsește în prima formă normală și UID-ul său este format dintr-un singur atribut atunci ea se găsește automat în a doua formă normală.

4. A treia formă normală

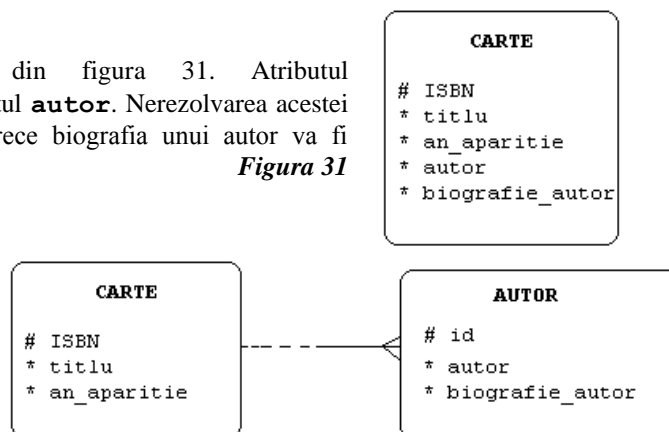
O entitate se găsește în a treia formă normală dacă și numai dacă se găsește în a doua formă normală și în plus nici un atribut care nu este parte a UID-ului nu depinde de un alt atribut non-UID. Cu alte cuvinte nu se acceptă dependențe tranzitive, adică un atribut să depindă de UID în mod indirect.

Luăm ca exemplu entitatea **CARTE** din figura 31. Atributul **biografie_autor** nu depinde de **ISBN** ci de atributul **autor**. Nerezolvarea acestei situații duce la memorarea de date redundante, deoarece biografia unui autor va fi memorată pentru fiecare carte scrisă de autorul respectiv.

Figura 31

Rezolvarea acestei situații este să creăm o nouă entitate **AUTOR**, pe care o legăm de entitatea **CARTE** printr-o relație 1:m

Figura 32



Atenție ! Acest model este corect doar dacă se acceptă că o carte are un singur autor.